Water Wave Packets

STEFAN JESCHKE, IST Austria and NVIDIA
CHRIS WOJTAN, IST Austria

This paper presents a method for simulating water surface waves as a displacement field on a 2D domain. Our method relies on Lagrangian particles that carry packets of water wave energy; each packet carries information about an entire group of wave trains, as opposed to only a single wave crest. Our approach is unconditionally stable and can simulate high resolution geometric details. This approach also presents a straightforward interface for artistic control, because it is essentially a particle system with intuitive parameters like wavelength and amplitude. Our implementation parallelizes well and runs in real time for moderately challenging scenarios.

CCS Concepts: • **Computing methodologies** → **Simulation by animation**; **Physical simulation**; *Real-time simulation*;

Additional Key Words and Phrases: water surface waves, liquid animation, particle system, wave packets, computational fluid dynamics, real-time simulation

ACM Reference format:
Stefan Jeschke and Chris Wojtan. 2017. Water Wave Packets. *ACM Trans. Graph.* 36, 4, Article 103 (July 2017), 12 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073678

1 INTRODUCTION

The motion of water surface waves is well-modeled by the two-phase incompressible Navier-Stokes equations. The general form of these equations is analytically and computationally intractable for detailed water surface geometry, so researchers traditionally apply a small-amplitude assumption that effectively linearizes the problem and restricts the waves to a height-field defined over a two-dimensional domain. This version of the water surface wave problem admits sinusoidal wave solutions that have a speed depending on their wavelength and water depth. However, although greatly simplified by the small-amplitude assumption, the problem is still too complex to solve analytically.

Research in computer graphics has approached this linearized water wave problem in a number of ways. Several methods have made progress by enforcing additional assumptions like shallow water [Kass and Miller 1990], infinite depth [Mastin et al. 1987; Tessendorf 2004b], omitting solid boundaries, or assuming static solid boundaries [Fournier and Reeves 1986; Jeschke and Wojtan 2015]. Other approaches use numerical techniques for solving partial differential equations in order to time-step through the water surface wave dynamics [Canabal et al. 2016; Tessendorf 2004a]. These approaches handle far more general scenarios, but they introduce nontrivial problems relating to stability, energy conservation, spatial resolution, and artistic control. Lastly, some methods approximate the waves themselves as Lagrangian particles [Yuksel et al. 2007].

This project has received funding from IST Austria and European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638176).
Authors' addresses: sjeschke@nvidia.com, wojtan@ist.ac.at.

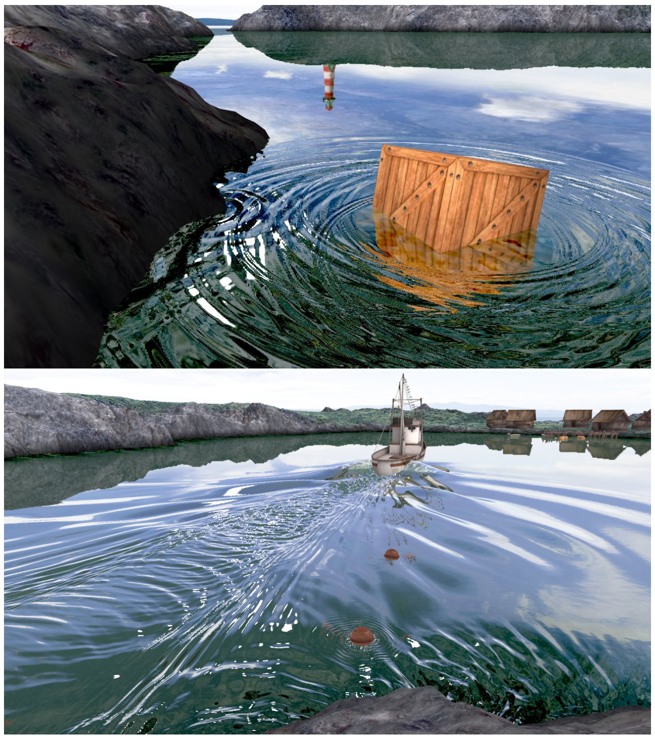


Fig. 1. We introduce a new water wave simulation algorithm inspired by wave packet theory. Our method can simulate accurate wave behaviors at real-time rates (top) and highly detailed wave scenarios offline (bottom).

This approach has the potential to handle very general scenarios with moving boundary geometry, but it produces solutions closer to those of a constant-speed wave equation, as opposed to a fully dispersive water wave equation.

We aim to leverage the potential of Lagrangian wave particles, but in a manner that plausibly simulates water wave dispersion. Instead of associating each particle with a single wave crest, we associate each particle with a *packet of wave energy* consisting of an entire spectrum of wavelengths and wave trains. We then describe how this wave packet moves and deforms to approximate the behavior of linearized water surface waves.

This paper makes the following contributions:

- **Wave packets** We introduce the concept of wave packets to computer graphics and describe their dynamics for dispersive water waves.
- **Visual detail** Our method improves upon previous Lagrangian particle approaches by exhibiting more visual detail (measured in wave crests per computational degree of freedom), and by incorporating qualitative wave behaviors like dispersion, diffraction, refraction, reflection, dissipation from the literature.

- **Efficient computation** The method is unconditionally stable, requires no artificial damping, is independent of grids or spatial resolution parameters, and is inherently parallel.
- **Novel control parameters** We introduce new mechanisms which allow artists to directly control wave spectra and computational complexity, exposing a straightforward trade-off between visual detail and computational speed.

2 RELATED WORK

2.1 Computer Graphics Literature

The animation of water surface waves has interested computer graphics researchers since at least 1980 [Schachter 1980]. As stated in the introduction, the main strategy since then has been to apply a multitude of assumptions to the Navier-Stokes equations in order to express the motion of the ocean in the form of sinusoidal waves [Hinsinger et al. 2002; Mastin et al. 1987; Tessendorf 2004b]. While such assumptions sacrifice the ability to simulate arbitrary fluid motion, they lead to extremely efficient computational methods. Subsequent work augmented these simple models with more interesting boundary conditions, splashes, spray, and breaking waves [Fournier and Reeves 1986; Gonzato and Le Saëc 1997; O'Brien and Hodgins 1995; Peachey 1986; Thuerey et al. 2007a,b; Ts'o and Barsky 1987]. The excellent survey by Darles et al. [2011] covers this ocean simulation literature in more detail.

Several novel approaches to water surface wave simulation have emerged in the past decade. Jeschke and Wojtan [2015] generalized the above analytical methods to handle complex boundaries while respecting wave behaviors like dispersion and diffraction. However, their method requires pre-computation and does not address moving boundaries. Instead of using the analytical Fourier solution, other researchers have explored two-dimensional Eulerian simulation [Tessendorf 2004a]. Subsequent research extended this method by addressing the accumulation of numerical errors over time [Tessendorf 2014] and more accurately capturing dispersive effects [Canabal et al. 2016]. Boundary-only approaches also show promise for efficiently simulating more general ocean wave behavior, though they currently require orders of magnitude more computation [Da et al. 2016; Keeler and Bridson 2014].

The "Wave Particles" approach of Yuksel et al. [2007] is most similar to ours. It represents each wave crest with its own set of particles, allowing wave reflections and interactions with dynamic objects. It is also straightforward to implement, parallelize, and control. Nevertheless, the one-particle-per-crest approach can be expensive when simulating long wave trains or high-frequency waves. Similarly, it is difficult for this method to simulate wave dispersion, because it adds a new dimension to the problem (one particle per crest, per wavelength). The method also implausibly transports wave energy at the phase speed instead of the group speed. Lastly, the approach does not address how to handle wave effects like refraction, diffraction, dispersion, or reflection off non-planar boundaries, although the subsequent dissertation [Yuksel 2010] provides useful insights on how to extend the method toward these aims. The Wave Particle approach has inspired follow-up work on background flows [Cords 2008], and the method's controllability and speed have made it an excellent candidate for simulating water in video games [Gonzalez-Ochoa 2016].

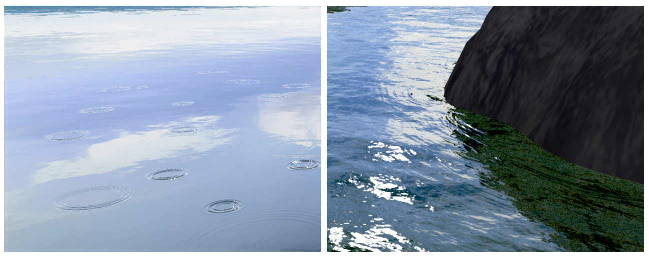


Fig. 2. Our method can efficiently add subtle detail like rain ripples (left) or boundary reflection waves (right) to existing scenes.

However one chooses to simulate surface water waves, the results can be used in many different applications. Previous researchers have used waves as boundary conditions or guide shapes [Nielsen and Bridson 2011; SideFX 2013], or as a type of physics-based procedural texture [Chentanez and Müller 2010]. Wave simulation parameters can also be tuned to achieve a desired look [Horvath 2015; Nielsen et al. 2013]. Researchers have also combined water surface wave simulations with fully three-dimensional simulations [Kim et al. 2013; Mercier et al. 2015; Thuerey et al. 2010; Yang et al. 2016; Yu et al. 2012]. We show how our method can augment some existing 2D simulations in Figure 2.

2.2 Physics Literature

The idea of considering Lagrangian water wave packets as a fundamental primitive, while novel to computer graphics, has a long history in theoretical physics. It seems to have originated during the explosion of theoretical quantum mechanics research in the early 20th century; in this case, the waves come from the Schrödinger equation [Birkhoff 1927]. Because the mathematical derivation of wave packets works for any dispersive equation, oceanographers have since used wave packet theory to also explain the transport of water wave energy [Pedlosky 2013]. Some even proposed to name the water wave packet the "hydron" and give it the same standing as other fundamental particles in physics, like photons and electrons [Synge 1962]. Despite the theoretical utility of considering water waves as packets, to the best of our knowledge, ours is the first method to use water wave packets as a fundamental primitive for numerical simulation.

3 WATER WAVE DYNAMICS

3.1 Airy wave theory

Airy wave theory [Airy 1841] describes a water surface as a height function that varies with time, $\eta(\mathbf{x}, t)$. We can use this framework to analyze how a group of waves propagates, and then use these theoretical results to develop a computationally efficient method for simulating water surface waves. We will first analyze the one-dimensional case, $\eta(x, t)$, and then extend the ideas to cover a two-dimensional water surface.

If we look at the Fourier transform of the water surface height field, we can view the surface as an integral of many different waves of varying wavelength:

$$\eta(x,t) = \int_{-\infty}^{\infty} a(k)\cos\left(kx - \omega(k,h)t\right) dk. \tag{1}$$

In this equation, k is the wavenumber, which is inversely proportional to the wavelength λ via the relationship $k = 2\pi/\lambda$, x is the spatial coordinate, h is the water depth, $\omega(k,h)$ is the angular frequency (which is inversely proportional to the period $T(k,h)$ via $\omega = 2\pi/T$), $a(k)$ is the amplitude of each wave, and t is time. The angular frequency has a special form that gives water waves their distinct characteristics:

$$\omega(k,h) = \sqrt{\left(gk + \frac{\sigma}{\rho}k^3\right)\tanh(kh)}, \tag{2}$$

where g is gravity, σ is the surface tension, and ρ is the water density. This relationship between the frequency ω and the wavenumber k is known as the *dispersion relation.* From this relation, we can see that the argument to the wave in Equation (1) can be factored into $k(x - c_p(k,h)t)$, where

$$c_p(k,h) = \frac{\omega(k,h)}{k} = \sqrt{\left(\frac{g}{k} + \frac{\sigma}{\rho}k\right)\tanh(kh)} \tag{3}$$

is the propagation speed of a given wavelength, known as the *phase velocity.* In 2D, the energy of a water wave with wavenumber k over a surface area A is

$$E(k) = \int_A \frac{1}{2}\left(\rho g + \sigma k^2\right)(a(\mathbf{x}))^2 dA. \tag{4}$$

Wave energy travels with the *group velocity* c_g, which is defined as

$$c_g(k) = \frac{d\omega}{dk} \tag{5}$$

and described further in Appendix A. In one dimension, k, c_p, and c_g are scalars. In two dimensions, they are vectors $\mathbf{k}$, $\mathbf{c}_p$, and $\mathbf{c}_g$, and we use the scalar k as shorthand for the wavevector's magnitude $||\mathbf{k}||$.

Intuition. We can extract some good intuition from the equations in this section if we consider their limit behaviors. For gravity waves ($\rho g \gg \sigma$) in deep water ($kh \gg 1$), longer wavelengths travel faster than shorter ones. We tend to see long waves on the outer edge of splashes, while shorter waves lag behind. Furthermore, the phase speed is equal to twice the group speed, so the wave crests actually outrun their energy and create the effect of wave crests disappearing at the outer edge of a splash. At intermediate water depth, c_p slows down as depth decreases, while c_g speeds up, until they finally become equal in shallow water ($kh \ll 1$), where all wavelengths travel at a constant speed $c_p = \sqrt{gh}$.

For capillary waves ($\rho g \ll \sigma$), which almost exclusively occur in the deep water/high wavenumber regime $kh \gg 1$, short wavelengths travel faster than longer ones. Thus, small splashes dominated by surface tension will have shorter wavelengths on the outer edge. Phase velocity is only two-thirds of the group velocity, so the energy outpaces the wave crests, creating the effect of waves materializing on the outer edge of a splash. All of these qualitative effects can be seen in our results.

3.2 Wave packets

In this paper, instead of computing with infinitely long wavetrains [Mastin et al. 1987] or single wave crests [Yuksel et al. 2007], we would like to propagate localized *packets* of waves. Each wave packet will represent a collection of similar wavelengths, and it will cover a larger region of space than a single wave crest. Such a strategy will allow us to simultaneously represent long wave trains with a single computational element, have the waves interact with a dynamically changing environment, and obey the qualitative behaviors described by Airy wave theory.

At this point in the derivation, we are free to choose what we'd like these wave packets to look like. We will list a few desired properties, so we can make some educated decisions later. First, we would like environmental variations like solid boundaries, water depth gradients, and user interactions to affect each packet as a whole. Thus, each wave packet should be compactly supported in *spatial* coordinates. On the other hand, fundamental physical properties like energy act locally in frequency space, so each wave packet should be compactly supported in *wavenumber* coordinates as well. So we want a kernel function $\phi(x)$ for our wave packets that acts locally in space, and whose Fourier transform $\Phi(k)$ also acts locally in wavenumber. Although no function can be truly topologically compact in both x and k due to the uncertainty principle for the Fourier Transform [Phillips 2005], many functions approximate this behavior by heavily weighting nearby values and falling off exponentially. We will use a Gaussian function $\phi(x) = \exp(-x^2)$ to represent the wave packet kernel, like many previous works in quantum physics [Liboff 2003].

Now, we can break up the integral in Equation (1) into a summation of individual packets of wavenumbers centered around some representative wavenumber k_j:

$$\eta(x,t) \approx \sum_{j=1}^{N} \int_{-\infty}^{\infty} \Phi(k - k_j)a(k)\cos\left(kx - \omega(k,h)t\right) dk, \tag{6}$$

where Φ is our wave packet shape and N is the number of wave packets. This equation is equal to Equation (1) if the Φ functions are a partition of unity (i.e., if $\sum_{j=1}^{n} \Phi(k - k_j) = 1$ for all k). Otherwise, as in our implementation, the right hand side is only approximate.

By assuming that the spectrum within each packet of waves is tightly concentrated around their representative wavelength k_j (i.e., if $\Phi(k)$ falls off quickly away from $k = k_j$), then we can approximate $w(k,h)$ and $a(k)$ with a first order Taylor expansion. After some analysis (carried out in Appendix B), we arrive at the expression

$$\eta(x,t) \approx \sum_{j=1}^{N} a_j \phi\left(x - c_g t\right)\cos\left(k_j(x - c_p t)\right), \tag{7}$$

which states that the water surface waves can be reasonably approximated by a sum of wave packets, each described by a kernel function ϕ which travels at *group speed* c_g and acts as an envelope for a single representative wave traveling at *phase speed* c_p. Each packet has a spatially-varying amplitude $a(x,t) = a_j\phi(x - c_g t)$, where a_j is a spatially-constant amplitude scale factor associated with packet j.

If we allow the speeds c_g and c_p to vary over time, then we should replace the analytical phase shifts $c_g t$ and $c_p t$ with integrated ones

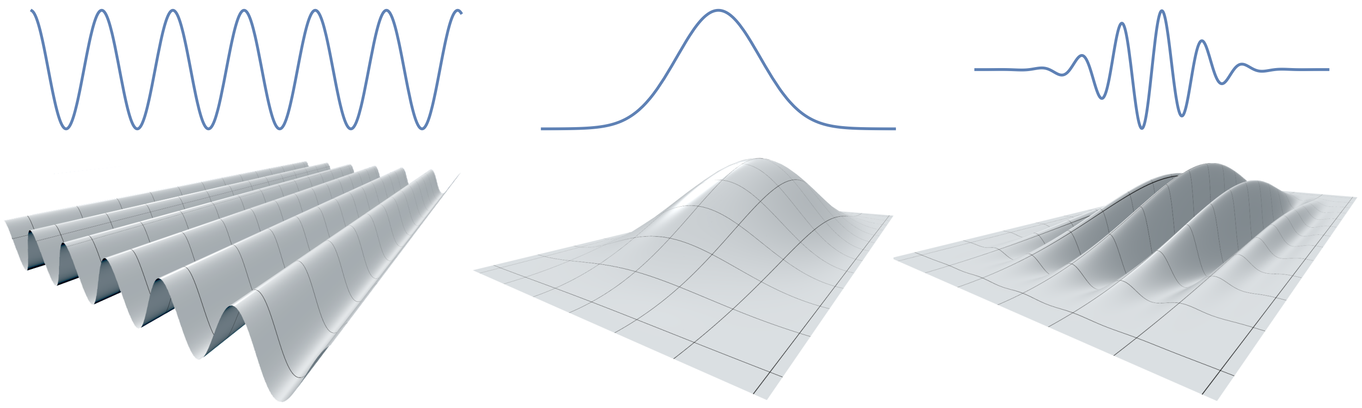


Fig. 3. The components of a single wave packet visualized in one dimension (top) and two dimensions (bottom). A cosine wave (left) is multiplied by a Gaussian envelope (center) to produce a wave packet (right).

$X_g(t) = \int c_g\,dt$ and $X_p(t) = \int c_p\,dt$, which build up a displacement over time by tracing along the paths of a packet and wave crest:

$$\eta(x,t) \approx \sum_{j=1}^{N} a_j \phi(x - X_g) \cos\big(k_j(x - X_p)\big). \tag{8}$$

We ultimately want our packets to be *Lagrangian* computational elements, so we prefer to express the wave dynamics in the packet's local coordinate frame $\hat{x} = x - X_g$:

$$\eta(\hat{x},t) \approx \sum_{j=1}^{N} a_j \phi(\hat{x}) \cos\big(k_j(\hat{x} - X_{pg})\big), \tag{9}$$

where $X_{pg} = \int (c_p - c_g)\,dt$ is the integrated drift between phase speed and group speed over time.

In two dimensions, this formula generalizes to

$$\eta(\hat{\mathbf{x}},t) \approx \sum_{j=1}^{N} a_j \phi(\hat{\mathbf{x}}) \cos\big(\mathbf{k}_j \cdot (\hat{\mathbf{x}} - \mathbf{X}_{pg})\big), \tag{10}$$

where bold notation indicates a two-dimensional vector. Note that $\mathbf{k}_j$ is now a wave vector with magnitude k_j, and $\mathbf{c}_p$ and $\mathbf{c}_g$ are velocity vectors with a direction and magnitude. We illustrate one term of this summation (a single wave packet) in Figure 3.

3.3 Qualitative wave behaviors

Reflection. A wave packet reflects when it collides with an obstacle. An elastic collision would perfectly preserve energy and keep the amplitude the same, and an inelastic collision can be simulated by decreasing the amplitude of the packet after a collision.

Dispersion. Some waves in the group travel faster than others, due to differing group velocities as a function of k. Consequently, the faster waves will push part of the packet ahead of the average group speed, and the slower waves will pull another part of the packet a bit behind. The result is that the packet spreads out as it traverses space, at a rate proportional to dc_g/dk. (This can be shown with a derivation similar to that in Appendix B that keeps higher order terms of Δk [Liboff 2003; Vandegrift 2004].) At the end of the day, this spreading stretches out our wave packet in the traveling direction, so conservation of energy dictates that the packet's amplitude must correspondingly decrease.

Refraction. The group speed of the packet may change as it traverses space, because c_g changes with water depth, and water depth can change over space. In these situations, the packet will change direction (refract), in a manner consistent with Snell's law [Breeding 1978].

Diffraction. Individual waves bend around obstacles, so wave packets will diffract in the same way. Because waves diffract differently depending on the wavenumber k, we believe the wave packet should spread out along the tangent direction. We have not yet worked out the theoretical diffractive spreading behavior and have not found it in the literature, but we know that the spreading and amplitude are constrained by the conservation of energy.

3.4 Energy of a wave packet

We would like to enforce conservation of energy as a wave packet propagates through space. The energy in Equation (4) is quadratic in amplitude and wavenumber and linear in area, so increasing the wavenumber or stretching the packet will correspondingly increase its energy. If we wish to keep energy unchanged, then we should alter the packet's amplitude to compensate for such changes. We calculate the exact amplitude scaling law for a packet with a Gaussian kernel in Appendix C.

In two spatial dimensions, waves not only traverse back and forth, but they can also spread apart. For example, a rain drop may start as a tight circle of wave energy, but the circumference increases as the waves travel outward. Conservation of energy dictates that the amplitude must decrease as each packet spreads out.

In nature, wave packets can *lose* energy due to a variety of factors. In addition to inelastic collisions described in the previous section, we also consider simplified energy dissipation due to viscosity, surface contamination, and more complicated non-linear behavior.

3.4.1 Viscosity. Viscosity is often modeled by explicitly reducing the wave amplitude in accordance with viscous potential flow theory [Padrino and Joseph 2007]. In the absence of all other amplitude changes, the amplitude will decay over time:

$$\frac{da}{dt} = -2\nu k^2 a \tag{11}$$

Although the viscosity of water ν is small (around $10^{-6}\text{m}^2/\text{s}$), the amplitude drops exponentially. The dependence on k^2 means that small wavenumbers (long wavelengths) basically ignore viscosity, while large wavenumbers are almost immediately damped out by it.

3.4.2 Surface contamination. In addition to viscosity, subtle surface contaminants (dirt, algae, oil etc.) will have a strong damping effect on water waves. This effect can be modeled as an additional decay rate [Dorrestein 1951; Le Méhauté 1988]

$$\frac{da}{dt} = -\frac{1}{2}\nu\left(\frac{\omega(k)}{2\nu}\right)^{\frac{1}{2}} ka \tag{12}$$

3.4.3 Non-linearity and breaking waves. Lastly, because Airy wave theory assumes small amplitudes ($a \ll \lambda$), it cannot accurately model steep waves, especially the energy dissipation that occurs when waves break and topple over. Defining *steepness* as the ratio of a wave's height divided by its wavelength, Dean and Dalrymple [1991] observe that waves in deep water break once their amplitude exceeds a steepness threshold of 7% of their wavelength, dissipating energy in the process. A simple way to approximate this type of dissipation is to reduce the amplitude of a wave packet until $a \leq 0.07\lambda$.

4 IMPLEMENTATION

Now that we have explained the theoretical behaviors of wave packets, we will explain how to implement them into an efficient algorithm for simulating detailed water wave behaviors. We model each wave packet completely independently, so they can be computed in parallel.

4.1 Representing wave packets

Each packet has a representative wavenumber $k_j = 2\pi/\lambda_j$ and an amplitude a_j. We model the spatial extent and deformation of the packet with a rectangular patch of initial dimensions $3\lambda_j$ in the traveling direction and $6\lambda_j$ in the tangential direction. Instead of using a single particle to track the position of the packet over time, we use two vertices $\mathbf{p}_1$ and $\mathbf{p}_2$ centered at the front edge of the packet. These two vertices allow us to track the deformation and rotation of the packet, which is important for simulating proper energy behavior, as well as refractions and reflections in a complicated environment. The distance between the two vertices will vary over time as waves focus and spread, so they are not kept at a fixed distance from each other. However, we subdivide the packet if they drift too far apart (Section 4.3). To model the extent of the packet in frequency space, we assign a range of wavenumbers (from k_{min} to k_{max}, with $k_j = (k_{\text{min}} + k_{\text{max}})/2$) to each packet.

Once we have an initial packet, we can model its dynamics. Starting with the initial travel direction, the two vertices propagate at the group speed c_g. We integrate the vertex positions through time with a simple forward Euler method.

$$\mathbf{p}(t + \Delta t) := \mathbf{p}(t) + \Delta t \mathbf{c}_g(\mathbf{p}(t), k_j) \tag{13}$$

More advanced time integration schemes may be useful for accurately resolving reflections and refractive angles, but they will not make the method any more stable. Our method is stable regardless of the time step size, due to our geometric method for conserving energy (described in Section 4.2).

We also model dispersion, by tracking the fastest and slowest wavenumbers in the packet's spectrum to see how far they drift apart. We assign each packet a length parameter l, which is initially set to $3\lambda_j$, and we track the dispersive stretching using a similar integration rule:

$$l(t + \Delta t) := l(t) + \Delta t(c_g(k_{\text{fast}}) - c_g(k_{\text{slow}})), \tag{14}$$

where k_{fast} and k_{slow} are the wavenumbers in the packet's range which correspond to the fastest and slowest group speeds, and c_g is the magnitude of $\mathbf{c}_g$. Conveniently, c_g depends straightforwardly on k: it monotonically decreases until $k = k_{\text{slowest}} \approx 143\text{m}^{-1}$, and then it monotonically increases. Thus, we can always assume the minimum and maximum wavelengths in the packet's range are the fastest and slowest, as long as we don't create any packets that span k_{slowest}. If a packet is created that *would* span k_{slowest}, then we carry out a dispersive subdivision routine (Section 4.3) and split the spectrum exactly at k_{slowest}. Specifically, we create two new packets, one with the part of the spectrum above k_{slowest}, and the other with the part less than k_{slowest}, and then delete the original packet.

We also change each packet's representative wavelength k_j depending on its environment, by ensuring that each wave group satisfies the dispersion relation (2) at all times. This constraint gives our simulator a mechanism to exhibit wave *shoaling*, i.e., when a wave slows down and decreases its wavelength as it enters shallow water. To do this, we first keep the angular frequency ω_j fixed for each packet. Then, during each time step, we solve for k_j by repeating $k_j := \omega_j / c_p(k_j, h)$ until convergence, as suggested by Jeschke and Wojtan [2015]. This scheme converges in very few iterations because it starts with a close initial guess. We believe we could optimize this function with a look-up table, but it is not currently a bottleneck.

Lastly, Equation (10) requires the computation of the integrated drift between phase speed and group speed for each packet, $\mathbf{X}_{pg} = \int (\mathbf{c}_p - \mathbf{c}_g)\, dt$. We again employ forward Euler integration:

$$\mathbf{X}_{pg}(t + \Delta t) := \mathbf{X}_{pg}(t) + \Delta t(\mathbf{c}_p(k_j) - \mathbf{c}_g(k_j)). \tag{15}$$

4.2 Amplitude adjustment

The energy scaling law described in Section 3.4 and Appendix C is exact for Gaussian wave packets in an infinite domain without internal boundaries, and we use it to approximate a more complicated domain.[1] We first compute the area of the packet in each time step

$$\begin{aligned} A_j(t_{n+1}) &= ||\mathbf{p}_1(t_{n+1}) - \mathbf{p}_2(t_{n+1})|| l(t_{n+1}) \\ A_j(t_n) &= ||\mathbf{p}_1(t_n) - \mathbf{p}_2(t_n)|| l(t_n) \end{aligned} \tag{16}$$

[1] Assuming that reflections will keep the peaks of wave packets outside of solid boundaries, the inaccuracies will only occur in the tails of the Gaussian ϕ function, so the total energy errors are exponentially small.

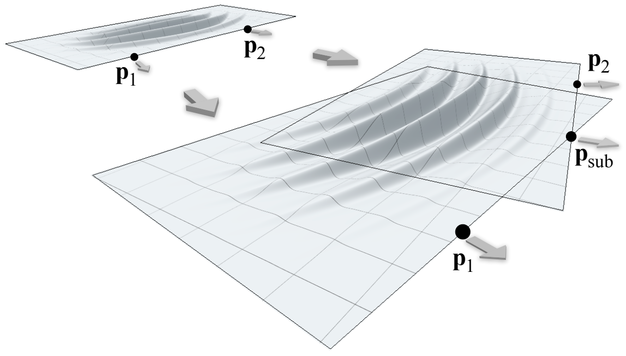


Fig. 4. During subdivision caused by tangential stretching, we subdivide the edge $\overline{\mathbf{p}_1\mathbf{p}_2}$ and distribute the new edges $\overline{\mathbf{p}_1\mathbf{p}_{\text{sub}}}$ and $\overline{\mathbf{p}_{\text{sub}}\mathbf{p}_2}$ to two new packets. This figure shows the result shortly after subdivision; the two new packets have already drifted apart, and the new line segments are no longer co-linear.

We then conserve energy by scaling the amplitude as described in Appendix C:

$$a_j(t_{n+1}) := a_j(t_n)\sqrt{\frac{(\rho g + \sigma k_n^2)A_j(t_n)}{(\rho g + \sigma k_{n+1}^2)A_j(t_{n+1})}} \tag{17}$$

where we temporarily introduced the notation $k_n := k_j(t_n)$ and $k_{n+1} := k_j(t_{n+1})$. For convenience, our implementation deletes any packet when its steepness falls below a minimum threshold.

We find this scheme for conserving energy extremely useful in practice. Its efficacy is independent of numerical parameters like the time step size and packet shape, unlike schemes derived by discretizing a differential equation (it is a *discrete* conservation law, rather than a *discretized* one). Our scheme also gives us a hard upper bound on the total wave energy in the system, and we argue in Appendix D that the packet's velocity c_g is bounded as well. These bounds, coupled with the fact that energy in each packet can only stay the same or dissipate, ensure that numerical blowups will never happen. Among other benefits, this unconditional numerical stability allows our waves to propagate at arbitrarily high speeds without a time step restriction or artificial damping. This behavior is particularly different from Eulerian methods which impose a stringent CFL condition for capillary waves.

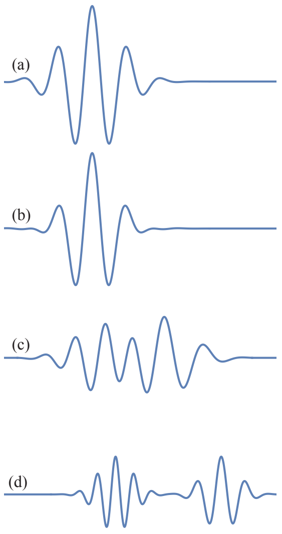
Extreme numerical integration errors (from Equation (13)), like missing a collision due to a huge time step, will still conserve energy, though they may generate packets with large areas and invisibly small amplitudes. The linear superposition principle prevents such erroneous packets from influencing any others.

4.3 Packet subdivision

When a packet deforms beyond a threshold, we choose to subdivide it in two. We can view this as a type of adaptive refinement strategy, to keep the space well-sampled with packets. We treat subdivision differently depending on whether excessive deformation is caused by geometric deformation (stretching in the tangential direction) or dispersion (stretching in the travel direction).

4.3.1 Geometric subdivision. We choose to subdivide a packet if the distance between the packet vertices $||\mathbf{p}_1(t_n) - \mathbf{p}_2(t_n)||$ exceeds $3\lambda_j$, or if the angle between the group velocities $\mathbf{c}_g(\mathbf{p}_1(t_n))$ and $\mathbf{c}_g(\mathbf{p}_2(t_n))$ exceeds 18 degrees. In either case, we replace the packet with two new overlapping ones by adding a new vertex $\mathbf{p}_{\text{sub}}$ at the position $(\mathbf{p}_1 + \mathbf{p}_2)/2$ and set $\mathbf{p}_1 = \mathbf{p}_{\text{sub}}$ for one of the new packets and $\mathbf{p}_2 = \mathbf{p}_{\text{sub}}$ for the other. Because each new packet has half the area of the original, each new packet gets half the energy of the original (each new amplitude is equal to the original divided by $\sqrt{2}$). All of the other parameters are kept constant during subdivision. Please see Figure 4 for an illustration of this process.

4.3.2 Dispersive subdivision. We also subdivide packets due to excessive dispersion, creating two new packets and distributing the packet's wave spectrum among them. When the packet length stretches beyond a threshold ($l > 1.3l_0$ in our implementation, where l_0 is the original length of the packet), we replace the packet by two new ones with exactly the same position and other packet parameters. We then distribute the wave spectrum to the new packets by introducing a mid-range wavenumber $k_{\text{sub}} = (k_{\min} + k_{\max})/2$ and setting $k_{\min} = k_{\text{sub}}$ and for one new packet and $k_{\max} = k_{\text{sub}}$ for the other. Afterwards, the representative wavenumbers of each new packet are set to $(k_{\min} + k_{\max})/2$. The energy distribution between the two new packets depends on the spectrum of the original packet (how much energy was allocated to each of the infinite number of wavenumbers in the original packet). Our implementation simply chooses new amplitudes such that energy is conserved and both new packets have the same steepness, implying that the original spectrum stored more energy in lower wavenumbers. The exact values of the new amplitudes are shown in Appendix E. To account for the accumulated packet spreading up to this point, we reset l for each packet to $l := (l - l_0)/2$. We provide a one-dimensional illustration of this dispersive subdivision in the inset figure. Here, an initial wave packet (a) is split into a sum of two packets with different wave spectra (b). The new packets then then drift apart due to dispersion (c), (d). The distance between packets in (d) is exaggerated for effect; our implementation will subdivide packets further before they drift this far apart.



4.4 Qualitative wave behaviors

The qualitative wave behaviors described in Section 3.3 are reproduced in our system with minimal additional work.

Reflection. When a wave packet vertex collides with a solid obstacle, we reflect each packet vertex as if it were a light ray reflecting off a reflective surface [Whitted 1980], specifically following the rules for reflecting line segments described by Jeschke and Wojtan [2015]. However, because packets have a finite spatial extent (a wavelet trailing behind the two packet vertices), reflecting all of the wave crests in the packet at once will cause a visually disturbing discontinuity. Instead, we continuously reflect all of the wave crests in the packet over time by creating a "ghost" packet which follows the original packet's trajectory before reflection, until it is completely absorbed into the obstacle.

Dispersion. As described in the previous section, we subdivide packets when they spread out due to internal dispersion. These subdivision events create packets with different representative wavenumbers. Because each of these new packets will have a different group speed, it naturally creates the effect of waves with different wavelengths traveling at different speeds.

Refraction. Each wave packet vertex also refracts as it travels. We compute the group speed at the beginning and end of each time step for each vertex, and we then compute the change in travel direction based on Snell's law. This process is similar to how it is done in previous work on wavefront tracking [Gamito and Musgrave 2002; Gonzato and Le Saëc 1997; Jeschke and Wojtan 2015; Ts'o and Barsky 1987]. Our implementation keeps the wavelet aligned with the packet traveling direction, effectively refracting both the envelope and the wavelet together. A more accurate method would refract both the envelope and the wavelet independently, but we did not find the subtle visual difference worth the additional computational expense and risk of numerical drift.

Diffraction. We implement diffraction in a manner identical to Jeschke and Wojtan [2015]: if one vertex of a packet collides with a solid boundary at a grazing angle, then we simply "glue" it to the boundary by restricting its motion to lie tangential to the solid boundary surface. This action is all that is needed to make waves diffract around obstacles. Highly curved boundaries will also stretch out packets, leading to rapid geometric packet subdivision (Section 4.3) and the expected exponential fall-off in amplitude [Levy and Keller 1959]. However, this diffractive approximation does not incorporate wavelength-dependence; more accurate diffraction requires further research.

Dissipation. We include the dissipative effects in Section 3.4 by analytically integrating these decay rates once per time step:

$$a(t+\Delta t) = a(t)\exp(-(2\nu k^2 + \nu k\,(\omega(k)/2\nu)^{\frac{1}{2}}\,/2)\,\alpha\Delta t), \tag{18}$$

where Δt is the time step size, and α is an optional control parameter described below in Section 5.3.

4.5 Visualization

To visualize the displacement field given by our wave packets, we begin with a flat surface and evaluate the wave height according to Equation (10). For evaluating this equation, we set $\mathbf{X}_g = (\mathbf{p}_1 + \mathbf{p}_2)/2$, making the local coordinates $\hat{\mathbf{x}}$ relative to the point at the front and center of the wave packet.

Although we use a Gaussian function for the kernel ϕ when computing the wave physics, we use a more compact and efficient approximation function ϕ_{viz} when evaluating (10) for visualization purposes. We first parameterize the rectangular patch representing each wave packet (Section 4.1) with coordinates $u, v \in [0,1]$, and then choose a simple cosine kernel that peaks at the center of the

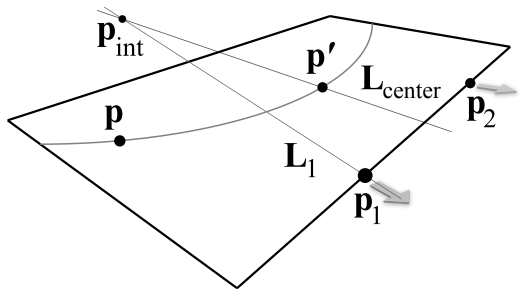


Fig. 5. This diagram shows how to evaluate circular wave arcs within a wave packet. Line $\mathbf{L}_{\text{center}}$ is parallel to the average traveling direction of both wave packet vertices and runs through the packet's geometric center, while line $\mathbf{L}_1$ is parallel to $\mathbf{c}_g(\mathbf{p}_1)$ and runs through vertex $\mathbf{p}_1$. $\mathbf{L}_{\text{center}}$ intersects $\mathbf{L}_1$ at $\mathbf{p}_{\text{int}}$. We map any point $\mathbf{p}$ to its corresponding point $\mathbf{p}'$ along the centerline with $\mathbf{p}' = \mathbf{p}_{\text{int}} + ||\mathbf{p} - \mathbf{p}_{\text{int}}||\mathbf{L}_{\text{center}}/||\mathbf{L}_{\text{center}}||$. This is where we evaluate $\hat{\mathbf{x}}$ in Equation (10).

packet and falls to zero with zero first derivative at the packet boundary:

$$\phi_{\text{viz}}(u,v) = (1-\cos(2\pi u))(1-\cos(2\pi v))/4 \tag{19}$$

We set the visual dimensions of the packet's rectangular patch to $3\lambda_j$ in the traveling direction and $6\lambda_j$ in the tangential direction.

The argument to the cosine function in Equation (10) maps spatial coordinates to a one-dimensional phase function. While this mapping is straightforward in 1D, we are left with several options for reducing a 2D rectangular packet to 1D. The simplest implementation assumes a piecewise-constant wavevector $\mathbf{k}_j$, which effectively keeps the crests of the cosine wavelets in Equation (10) in straight parallel lines. However, we found that a piecewise circular-approximation looks far more realistic, especially near boundaries and sources that tend to emanate circular waves. To achieve this, we follow the geometric construction in Figure 5. We show a comparison between using a constant $\mathbf{k}_j$ and a circular one in Figure 6.

Subdividing a packet into two new ones (Section 4.3) can create visual popping artifacts without very conservative subdivision thresholds. To allow more efficient computation without visual artifacts, we visually fade between the original packet and the new subdivided ones as they propagate over a distance of $3\lambda_j$. We also visually blend reflecting packets, during the period when the "ghost" packet penetrates an obstacle and the reflected packet emerges from it. We do this because, although the continuous reflection approach in Section 4.4 is perfect for planar solid obstacles, large rectangular packets can appear to erroneously leak around the corners of obstacles that are highly curved.

We rapidly evaluate the wave heights using a GPU-accelerated level of detail approach similar to Hinsinger et al. [2002] and Jeschke and Wojtan [2015]. We first create a pixel grid in the viewport and then project the pixel locations onto the plane representing the water domain. At each of these sample points, we use GPU acceleration to evaluate η in Equation (10). Our method also allows additional horizontal displacements (with a Gerstner or Biesel model [Fournier and Reeves 1986], for example), but we did not find this necessary. We also found it useful to add additional ocean texture to some

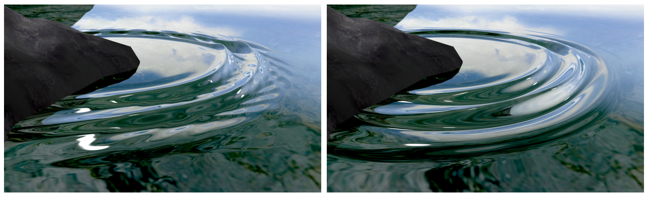


Fig. 6. Overlapping packets consisting of linear wavefronts can produce cross-hatching artifacts in high curvature regions (left). Circular arc wavefronts remove these artifacts (right).

scenes by evaluating a Fourier ocean spectrum [Tessendorf 2004b] and a simple foam shader on top of our simulated waves.

5 CONTROL

Similar to any particle system, we can control the look and feel of the results by selectively choosing initial parameters. We can also control the computational complexity by customizing the "lifetime" of each packet.

5.1 Initializing wave packets

Although the wave packet parameters allow plenty of room for artistic control, all of our examples use the same rules for initializing waves. We tune the initial wave spectrum by hand (a simple rule of thumb is to estimate the lowest frequency of the disturbance causing the waves, and then fill in the higher frequencies with noise at a lower amplitude). We choose the initial position of each packet based on the shape of the initial disturbance (like dividing up the outline of a boat or small circle into individual packets). We set the initial traveling direction normal to the starting shape, with the magnitude of $\mathbf{c}_g$ determined by Equation (5).

In our examples, we emit waves from an initially circular shape, or from solid obstacles like boats and buoys. We can also change the apparent roughness of an obstacle's surface by altering the spectrum of waves that reflect off it; stretching the reflected spectrum to higher wavenumbers seems to visually indicate a more detailed surface. To implement elementary one-way fluid-to-solid coupling, we pull the solid toward the water surface at each time step with a buoyancy force integrated with an unconditionally stable backward Euler integration scheme. We did not yet implement coupling from moving objects to waves, so our results exhibit inaccurate reflections near floating objects.

5.2 Wakes

Once we know how to emit waves from a source shape at a single instant in time, it is straightforward to simulate a continuous source (like a moving boat) by repeatedly emitting waves each time step. However, the characteristic "wake" shape that we expect to see is quite expensive to simulate using this straightforward technique. The difficulty arises from interference: although a huge number of waves are simulated, most of them will destructively interfere and provide no visual feedback. Instead, we would like to emit *only* the visually dominant waves in the wake pattern, but the trick is to figure out which ones. We turn to Kelvin's theory of wakes for this information [Johnson 1997; Thomson 1891; Whitham 2011].

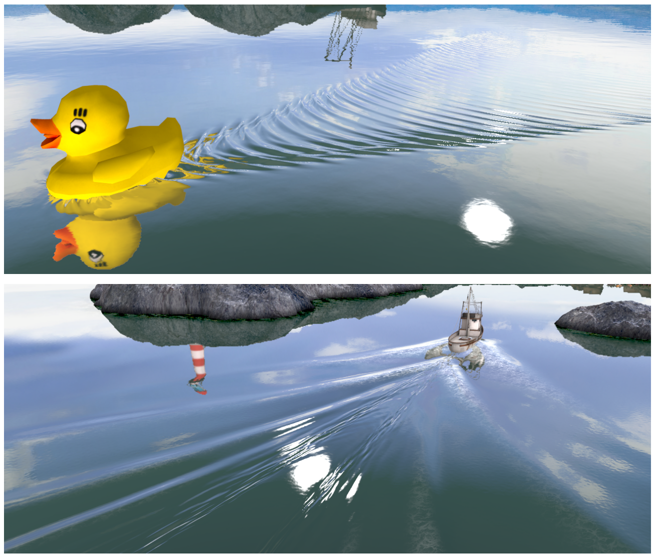


Fig. 7. Different values of $\mathbf{v}$ produce qualitatively different wakes. Slow disturbances (top) emphasize connected circular arcs, while fast motions (bottom) emphasize long chevron patterns.

Kelvin showed that, for a circular wave source moving at a constant velocity in the deep water gravity wave regime, the constructively interfering waves follow the relationship

$$k = \frac{g}{||\mathbf{v}||^2 \cos^2\theta}, \tag{20}$$

where $\mathbf{v}$ is the velocity of the wave source, $\theta \in [-\pi/2, \pi/2]$ is the angle that the wave packet's travel direction makes with $\mathbf{v}$, and k is the dominant wavenumber in that direction. The strongest wave packet in this family occurs at $\theta \approx 35.3°$, which corresponds to the outer edge of the wake. When we simulate a moving boat, we emit a small number of packets each time step at random angles with representative wavenumbers that obey Equation (20). Figure 7 shows an example of different wakes that our method can generate using this strategy.

5.3 Control over packet lifetime

In addition to deleting packets when their amplitude is too small (Section 4.2), we can delete them more aggressively if we want to control the algorithm run-time. Instead of naïvely assigning a "lifetime" to each wave packet, we adaptively scale the existing physical damping mechanism. Each time step, we re-compute a control parameter α to reduce the current number of wave packets N down to a target number of surviving packets N_{target}, as follows:

$$\alpha = 1 + \beta \max(0, N/N_{\text{target}} - 1)^2 \tag{21}$$

with stiffness parameter $\beta = 100$ for all of our experiments. We then substitute α into the damping exponent, as in Equation (18). This strategy effectively implements a soft constraint on the maximum number of wave packets, because there is no feedback on how many packets survived the new α parameter until the next time step.

6 RESULTS AND DISCUSSION

We have tested our method across a wide range of parameters (surface tension, viscosity, gravity) and challenging environments (large and small scales, varying initial wave spectra, varying water depth, and complex boundaries). We show some examples of our method in Figures 1 and 2. Please see our supplemental video for more examples.

Our method is inspired by physical laws, and it exactly satisfies energy conservation, wave propagation speeds, and damping by construction. Because we explicitly model energy and group velocity, our approach also recreates subtle behaviors like the disappearance of wave crests as they enter low-energy regions. This effect is extremely expensive to model by interference alone.

6.1 Parameters

Most of the parameters in our model (surface tension, viscosity, gravity, etc.) map directly to measurable physical quantities. However, the method also has numerical parameters like the thresholds for subdividing packets and the minimum steepness threshold for deleting packets. These parameters do not affect the qualitative behavior of our simulations, but they do affect performance and visual detail. Varying the subdivision thresholds causes earlier or later subdivision, and varying the minimum steepness threshold shortens or extends packet lifetimes. We chose these thresholds empirically to trade off between visual detail and the number of packets.

6.2 Limitations

We intentionally neglected some minor effects, like a viscosity-dependent wave speed or independent refraction effects for wavelets and groups. However, the main limitations of our method come from the oversimplified linear theory used to derive it. All non-linear effects, like the facts that wave speeds should technically depend on amplitude and colliding waves do not strictly obey the superposition principle, are absent from our results. More extreme effects, like breaking waves and topology changes during splashes, are completely outside the scope of our method. We also do not know how to make our method feed back on itself and alter its own liquid domain, like when a tidal wave rolls ashore and spreads water to previously dry areas.

Lastly, we do not yet have a satisfying theory for seeding wave packets (or wave particles, for that matter). We have presented a theoretical model for seeding wakes, but we would like to have a general method for computing the initial wave spectrum.

6.3 Efficiency

We implemented the wave packet dynamics in parallel on the CPU. Our method's run-time depends approximately linearly on the number of simulated wave packets, as illustrated by Figure 8. On our test machine (Laptop with 4-core 2.6GHz Intel i7-6700HQ, 32GB RAM, GeForce GTX 1070 GPU), a single packet takes about 2×10^{-4} milliseconds to simulate per time step. The 60fps simulation speed cut-off was about 85k wave packets, although our code could benefit from additional optimization. The interactive simulation in Figure 1 (top) used around 50k packets with a total frame rate consistently above 20fps; rendering is the bottleneck of our implementation, taking 2/3 of the total time. The most complex example at the end of our supplementary video used about 6 million wave packets and took between one and two seconds per frame to simulate. In the future, we plan to achieve a considerable speed-up by porting our wave packet code to the GPU.

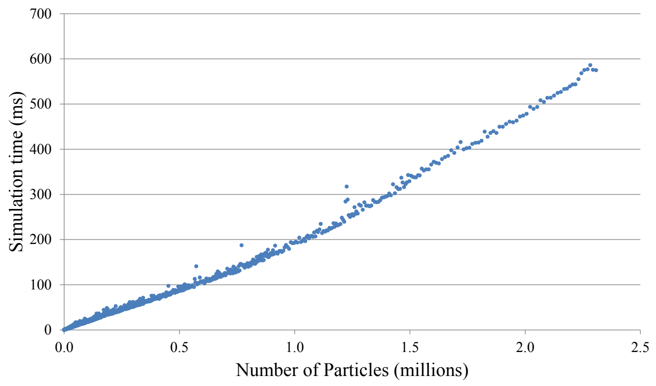


Fig. 8. We recorded the number of wave packets and the simulation time per time step in a complicated simulation (many independent splash events near complex boundaries and varying water depth). This plot shows a near linear relationship between the two.

6.4 Comparison to alternative methods

Figure 9 and our supplementary video compares our method to an implementation of Wave Particles [Yuksel et al. 2007]. Like ours, the Wave Particles approach is a Lagrangian technique that propagates wave information through the environment. Although it has more computational expense per element (per packet in our case, per particle in Wave Particles), we argue that our method is more versatile and physically plausible. Wave particles cannot simulate dispersion or realistic energy propagation, which we believe are important for a visually plausible simulation. These artifacts can be seen in our video when wave particles drift apart and leave erroneous gaps between them, and when wave crests fail to disappear as they run to the edge of a splash wave. Furthermore, our implementation allocates a $3\lambda_j \times 6\lambda_j$ area of wavefunction detail to each packet, representing approximately 3×6 isotropic wave crest samples. Consequently, we need around 18 wave particles to represent the same level of detail as a single wave packet.

It is difficult to compare our method directly to an Eulerian method [Tessendorf 2004a]. Eulerian methods tend to handle qualitative wave effects like dispersion, reflection, and diffraction without any of the additional implementation overhead that is required by Lagrangian approaches like ours. In particular, reflection and diffraction fall out naturally from Eulerian approaches by simply adding boundary conditions. However, the standard numerical difficulties with Eulerian methods, like the CFL condition imposing a maximum stable time step, Nyquist's limit imposing a minimum visible wavelength, and discretization errors causing artificial viscosity, are not present in our method. Our method can stably simulate arbitrarily high wavenumbers, large surface tension forces, and large time steps, without encountering instability or numerical dissipation.

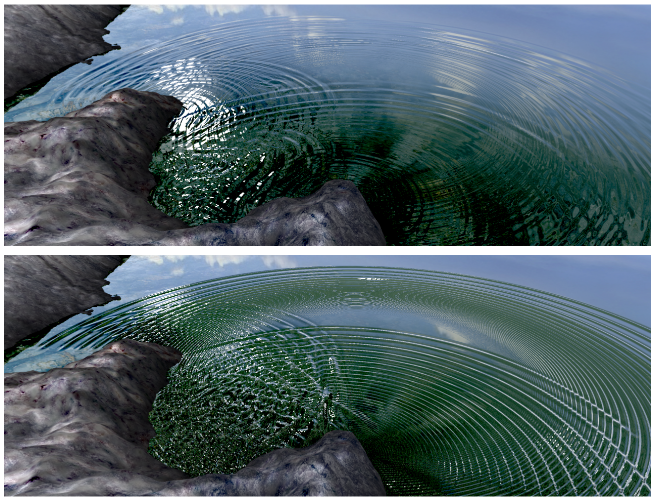


Fig. 9. We compare our method (top) to an augmented version of Wave Particles that incorporates dispersion, dissipation, diffraction, and reflection off curved obstacles (bottom). Even with these extensions, the wave particles approach exhibits unrealistic phase and group speeds, and it creates gaps when wave crests separate. These images come from different times in the simulations, because wave particles overestimate the wave speed.

Although the theory behind wave packets may be cumbersome, the implementation is quite simple, easy to parallellize, and robust. Source code for our implementation is available at: https://doi.org/10.5281/zenodo.525184 and http://pub.ist.ac.at/group_wojtan/projects/2017_Jeschke_WaterWavePackets/

ACKNOWLEDGEMENTS

We wish to thank everyone who contributed to this paper, especially the anonymous reviewers, Camille Schreck and David Hahn for proof-reading the paper, and Reiner Prokein for creating the models used in our examples. This research was supported by the Scientific Service Units (SSU) of IST Austria through resources provided by Scientific Computing. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638176).

A GROUP VELOCITY

Group velocity is defined as $c_g = d\omega/dk$, with ω given by Equation (2). This can be computed exactly

$$c_g = \frac{h\left(gk + \frac{\sigma}{\rho}k^3\right)\operatorname{sech}^2(kh) + \left(g + 3\frac{\sigma}{\rho}k^2\right)\tanh(kh)}{2\omega} \tag{22}$$

or approximated with finite differences, for example

$$c_g \approx \frac{\omega(k+\Delta k) - \omega(k)}{\Delta k}, \tag{23}$$

with Δk equal to a small value like 10^{-4}. We use the exact version for the results in this paper.

B WAVE PACKET DERIVATION

For convenience of notation, we begin with Equation (1) in exponential form:

$$\eta(x,t) = \int_{-\infty}^{\infty} a(k) e^{i(kx-\omega(k)t)} dk. \tag{24}$$

A Taylor series expansion tells us that a group of waves centered around some wavenumber k_j will have a dispersion relation approximately equal to

$$\omega(k_j + \Delta k_j, h) \approx \omega_j + c_g(k_j, h)\Delta k_j, \tag{25}$$

where we use the notation $\Delta k_j = k - k_j$, $c_g(k_j, h) = \frac{d\omega}{dk}(k_j, h)$, and we use the shorthand $\omega_j = \omega(k_j, h)$. We similarly expand $a(k)$ about k_j to get

$$a(k_j + \Delta k_j) \approx a_j + \frac{da}{dk}(k_j)\Delta k_j, \tag{26}$$

where we use the notation $a_j = a(k_j)$. Plugging these results into Equation (6) gives

$$\eta(x,t) = \sum_{j=1}^{N} a_j \int_{-\infty}^{\infty} \Phi(\Delta k_j) e^{i\left(kx - (\omega_j + c_g \Delta k_j)t\right)} d\Delta k_j + O(\Delta k_j^2).$$

Since the wave groups are locally concentrated about k_j, Δk_j is small. We neglect the higher order terms to get

$$\begin{aligned}
\eta(x,t) &\approx \sum_{j=1}^{N} a_j \int_{-\infty}^{\infty} \Phi(\Delta k_j) e^{i\left(kx - (\omega_j + c_g \Delta k_j)t\right)} d\Delta k_j \\
&= \sum_{j=1}^{N} a_j \int_{-\infty}^{\infty} \Phi(\Delta k_j) e^{i\left(k_j x + \Delta k_j x - \omega_j t - c_g \Delta k_j t\right)} d\Delta k_j \\
&= \sum_{j=1}^{N} a_j \int_{-\infty}^{\infty} \Phi(\Delta k_j) e^{i\left(k_j x - \omega_j t\right)} e^{i\left(\Delta k_j x - c_g \Delta k_j t\right)} d\Delta k_j \\
&= \sum_{j=1}^{N} a_j e^{ik_j(x - c_p t)} \int_{-\infty}^{\infty} \Phi(\Delta k_j) e^{i\Delta k_j (x - c_g t)} d\Delta k_j. \qquad (27)
\end{aligned}$$

The shift property of the inverse Fourier transform gives us

$$\eta(x,t) \approx \sum_{j=1}^{N} a_j e^{ik_j(x-c_p t)} \phi(x - c_g t) \tag{28}$$

or, taking only the real part:

$$\eta(x,t) \approx \sum_{j=1}^{N} a_j \cos\left(k_j(x - c_p t)\right) \phi\left(x - c_g t\right). \tag{29}$$

C PACKET ENERGY

We begin with a wave packet with amplitude $a(\hat{\mathbf{x}}) = a\,\phi(\hat{\mathbf{x}})$. Equation (4) tells us that a packet has the following energy for each wavelength:

$$\begin{aligned}
E &= \iint_{\mathbb{R}^2} \frac{1}{2}\left(\rho g + \sigma k^2\right)\left(a(\hat{\mathbf{x}})\right)^2 \, d\hat{x}\, d\hat{y} \\
&= \frac{1}{2}\left(\rho g + \sigma k_j^2\right) a^2 \iint_{\mathbb{R}^2} \phi^2(\hat{\mathbf{x}}) \, d\hat{x}\, d\hat{y}. \qquad (30)
\end{aligned}$$

Assuming a Gaussian kernel $\phi(\hat{x},\hat{y}) = \exp\left(-(\hat{x}/s_x)^2 - (\hat{y}/s_y)^2\right)$, where s_x is a scale factor in the $\hat{x}$- direction, and s_y is a scale factor in the $\hat{y}$- direction,

$$\iint_{\mathbb{R}^2} \phi^2(\hat{\mathbf{x}})\, d\hat{x}\, d\hat{y} = \iint_{\mathbb{R}^2} e^{-2\left((\hat{x}/s_x)^2 - (\hat{y}/s_y)^2\right)}\, d\hat{x}\, d\hat{y}$$
$$= \frac{\pi}{4} s_x s_y = \frac{\pi}{4} A,$$

where $A = s_x s_y$ is an area scale factor relative to the unscaled packet, which we can think of as the packet's current area. If we wish for the packet to conserve energy from time t_n to time t_{n+1}, then we can set the two energies equal. For clarity, we will temporarily use the subscript to indicate which time step is used to evaluate each quantity, so a_n is the packet's amplitude at time step n.

$$\frac{1}{2}\left(\rho g + \sigma k_n^2\right) a_n^2 \frac{\pi}{4} A_n = \frac{1}{2}\left(\rho g + \sigma k_{n+1}^2\right) a_{n+1}^2 \frac{\pi}{4} A_{n+1} \tag{31}$$

Solving for the amplitude at t_{n+1} gives us:

$$a_{n+1} = a_n \sqrt{\frac{\left(\rho g + \sigma k_n^2\right) A_n}{\left(\rho g + \sigma k_{n+1}^2\right) A_{n+1}}} \tag{32}$$

D BOUND ON PACKET VELOCITY

Here we argue that the group speed c_g is bounded, so wave packets will not blow up to arbitrarily large velocities. The group speed from Equation (22) is maximized in the deep water limit ($kh \rightarrow \infty$):

$$\lim_{kh\to\infty} c_g = \frac{g + 3\frac{\sigma}{\rho} k_j^2}{2\sqrt{g k_j + \frac{\sigma}{\rho} k_j^3}} \tag{33}$$

This expression for the largest possible packet velocity is finite as long as the packet's representative wavenumber k_j is neither zero nor infinite in deep water. Our simulator defines k_j as the limit of the fixed point iteration $k_j := \omega_j / c_p(k_j, h)$ (Section 4.1), and we keep ω_j fixed, so a finite c_p in the deep water regime would imply a finite k_j and a bounded group velocity.

To show that k_j is finite, we first notice that the deep water phase speed

$$\lim_{kh\to\infty} c_p = \sqrt{\left(\frac{g}{k_j} + \frac{\sigma}{\rho} k_j\right)} \tag{34}$$

is always greater than zero, so k_j is bounded from above in deep water. Next, we note that c_p increases with depth,

$$\frac{dc_p}{dh} = k_j \operatorname{csch}(2k_j h) \sqrt{\tanh(k_j h)\left(\frac{g}{k_j} + \frac{\sigma}{\rho} k_j\right)} > 0 \tag{35}$$

so it cannot drop to zero as it enters deep water. Therefore, because k_j is finite, the maximum packet velocity is bounded.

E DISTRIBUTING ENERGY DURING PACKET SUBDIVISION

As described in Section 4.3, we must assign amplitudes to two new packets after subdivision such that energy is conserved. Using Equation (4) to compute energy gives us the constraint equation

$$(\rho g + \sigma k_0^2) a_0^2 = (\rho g + \sigma k_1^2) a_1^2 + (\rho g + \sigma k_2^2) a_2^2, \tag{36}$$

where a_0 and k_0 are the amplitude and representative wavenumber of the original packet, and a_1, a_2 and k_1, k_2 the amplitudes and representative wavenumbers of the new packets after subdivision. This gives us one constraint with two unknowns, a_1 and a_2.

The other constraint on these amplitudes will in principle depend on the continuous wave spectrum contained in the original packet. Our implementation does not store entire spectra per packet, so we make an arbitrary assumption (based on what we thought looked nice) in order to make progress. We assume that the steepness of each new packet is equal:

$$a_1 k_1 = a_2 k_2. \tag{37}$$

Solving the system of these two equations gives us

$$a_1 := a_0 k_2 \sqrt{\frac{\rho g + \sigma k_0^2}{\rho g (k_1^2 + k_2^2) + 2\sigma k_1^2 k_2^2}} \tag{38}$$
$$a_2 := a_1 k_1 / k_2. \tag{39}$$

Or, for environments where surface tension is negligible:

$$a_1 := \frac{a_0 k_2}{\sqrt{k_1^2 + k_2^2}} \tag{40}$$
$$a_2 := a_1 k_1 / k_2. \tag{41}$$

REFERENCES

George Biddell Airy. 1841. *Tides and waves.* London.

GD Birkhoff. 1927. THE FOUNDATION OF QUANTUM MECHANICS. *Bull. Amer. Math. Soc.* (1927).

J Ernest Breeding. 1978. Velocities and refraction laws of wave groups: A verification. *Journal of Geophysical Research: Oceans* 83, C6 (1978), 2970–2976.

José A. Canabal, David Miraut, Nils Thuerey, Theodore Kim, Javier Portilla, and Miguel A. Otaduy. 2016. Dispersion Kernels for Water Wave Simulation. *ACM Trans. Graph.* 35, 6, Article 202 (Nov. 2016), 10 pages. DOI: https://doi.org/10.1145/2980179.2982415

Nuttapong Chentanez and Matthias Müller. 2010. Real-time simulation of large bodies of water with small scale details. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.* 197–206.

Hilko Cords. 2008. Moving with the flow: Wave particles in flowing liquids. In *Winter School of Computer Graphics (WSCG).*

Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-only liquids. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 78.

Emmanuelle Darles, Benoît Crespin, Djamchid Ghazanfarpour, and Jean-Christophe Gonzato. 2011. A survey of ocean simulation and rendering techniques in computer graphics. In *Comput. Graph. Forum*, Vol. 30. 43–60.

Robert George Dean and Robert A Dalrymple. 1991. *Water wave mechanics for engineers and scientists.* World Scientific.

R Dorrestein. 1951. General linearized theory of the effect of surface films on water ripples. *Nederl. Akad. Van Wetenschapen B* 54 (1951), 250–272.

Alain Fournier and William T Reeves. 1986. A simple model of ocean waves. In *Computer Graphics*, Vol. 20. ACM, 75–84.

Manuel N Gamito and F Kenton Musgrave. 2002. An accurate model of wave refraction over shallow water. *Computers & Graphics* 26, 2 (2002), 291–307.

Carlos Gonzalez-Ochoa. 2016. Advances in Real-Time Rendering in Games: Rendering Rapids in Uncharted 4. *ACM SIGGRAPH Courses* (2016).

Jean-Christophe Gonzato and Bertrand Le Saëc. 1997. A phenomenological model of coastal scenes based on physical considerations. In *Computer Animation and Simulation '97.* 137–148.

Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. 2002. Interactive animation of ocean waves. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* 161–166.

Christopher J Horvath. 2015. Empirical directional wave spectra for computer graphics. In *Proceedings of the 2015 Symposium on Digital Production.* ACM, 29–39.

Stefan Jeschke and Chris Wojtan. 2015. Water Wave Animation via Wavefront Parameter Interpolation. *ACM Trans. Graph.* 34, 3, Article 27 (May 2015), 14 pages. DOI: https://doi.org/10.1145/2714572

R.S. Johnson. 1997. *A modern introduction to the mathematical theory of water waves.* Vol. 19. Cambridge university press.

M. Kass and G. Miller. 1990. Rapid, stable fluid dynamics for computer graphics. In *Computer Graphics*, Vol. 24. 49–57.

Todd Keeler and Robert Bridson. 2014. Ocean Waves Animation using Boundary Integral Equations and Explicit Mesh Tracking. In *Proceedings of the 13th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '14)*. Eurographics.

Theodore Kim, Jerry Tessendorf, and Nils Thuerey. 2013. Closest Point Turbulence for Liquid Surfaces. *ACM Trans. Graph.* 32, 2, Article 15 (April 2013), 13 pages. DOI: https://doi.org/10.1145/2451236.2451241

Bernard Le Méhauté. 1988. Gravity–capillary rings generated by water drops. *Journal of Fluid Mechanics* 197 (1988), 415–427.

Bertram R Levy and Joseph B Keller. 1959. Diffraction by a smooth object. *Comm. Pure and Appl. Math.* 12, 1 (1959), 159–209.

Richard L Liboff. 2003. *Introductory quantum mechanics*. Addison-Wesley.

Gary A Mastin, Peter A Watterberg, and John F Mareda. 1987. Fourier synthesis of ocean scenes. *Computer Graphics and Applications, IEEE* 7, 3 (1987), 16–23.

Olivier Mercier, Cynthia Beauchemin, Nils Thuerey, Theodore Kim, and Derek Nowrouzezahrai. 2015. Surface turbulence for particle-based liquid simulations. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 202.

M.B. Nielsen and R. Bridson. 2011. Guide shapes for high resolution naturalistic liquid simulation. In *ACM Trans. Graph.*, Vol. 30. ACM, 83.

Michael B Nielsen, Andreas Söderström, and Robert Bridson. 2013. Synthesizing waves from animated height fields. *ACM Trans. Graph.* 32, 1 (2013), 2.

James F O'Brien and Jessica K Hodgins. 1995. Dynamic simulation of splashing fluids. In *Proc. Comp. Anim. '95*. IEEE, 198–205.

Juan C Padrino and Daniel D Joseph. 2007. Correction of Lamb's dissipation calculation for the effects of viscosity on capillary-gravity waves. *Physics of Fluids* 19 (2007), 082105.

Darwyn R Peachey. 1986. Modeling waves and surf. In *Computer Graphics*, Vol. 20. ACM, 65–74.

Joseph Pedlosky. 2013. *Waves in the ocean and atmosphere: introduction to wave dynamics*. Springer Science & Business Media.

T Phillips. 2005. The mathematical uncertainty principle. *Monthly Essays on Mathematical Topics* (2005).

Bruce Schachter. 1980. Long crested wave models. *Computer Graphics and Image Processing* 12, 2 (1980), 187–201.

SideFX. 2013. Houdini 13.0 Wave Layer Tank. (December 2013). http://www.sidefx.com/docs/houdini13.0/shelf/wavelayertank.

JL Synge. 1962. Water waves and hydrons. *Science* 138, 3536 (1962), 13–15.

Jerry Tessendorf. 2004a. Interactive water surfaces. *Game Programming Gems* 4 (2004), 265–274.

Jerry Tessendorf. 2004b. Simulating ocean water. *ACM SIGGRAPH Courses* (2004).

Jerry Tessendorf. 2014. *eWave: Using an Exponential Solver on the iWave Problem.* . Technical Note.

William "Lord Kelvin" Thomson. 1891. *Popular lectures and addresses*. Vol. 3. Macmillan London. 481–8 pages.

Nils Thuerey, Matthias Muller-Fischer, Simon Schirm, and Markus Gross. 2007a. Real-time breaking waves for shallow water simulations. In *Proc. Pacific Graphics*. IEEE, 39–46.

Nils Thuerey, F. Sadlo, S. Schirm, Matthias Müller-Fischer, and Markus Gross. 2007b. Real-time Simulations of Bubbles and Foam Within a Shallow Water Framework. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.* 191–198. http://dl.acm.org/citation.cfm?id=1272690.1272716

N. Thuerey, C. Wojtan, M. Gross, and G. Turk. 2010. A multiscale approach to mesh-based surface tension flows. *ACM Trans. Graph.* 29, 4 (2010), 48.

Pauline Y Ts'o and Brian A Barsky. 1987. Modeling and rendering waves: wave-tracing using beta-splines and reflective and refractive texture mapping. *Computer Graphics* 6, 3 (1987), 191–214.

Guy Vandegrift. 2004. The diffraction and spreading of a wavepacket. *American Journal of Physics* 72, 3 (2004), 404–407.

Gerald Beresford Whitham. 2011. *Linear and nonlinear waves*. Vol. 42. John Wiley & Sons.

Turner Whitted. 1980. An Improved Illumination Model for Shaded Display. *Commun. ACM* 23, 6 (June 1980), 343–349. DOI: https://doi.org/10.1145/358876.358882

Sheng Yang, Xiaowei He, Huamin Wang, Sheng Li, Guoping Wang, Enhua Wu, and Kun Zhou. 2016. Enriching SPH simulation by approximate capillary waves. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 29–36.

Jihun Yu, Chris Wojtan, Greg Turk, and Chee Yap. 2012. Explicit Mesh Surfaces for Particle Based Fluids. *EUROGRAPHICS 2012* 30 (2012), 41–48.

Cem Yuksel. 2010. *Real-time water waves with wave particles*. Ph.D. Dissertation. Citeseer.

Cem Yuksel, Donald H House, and John Keyser. 2007. Wave particles. *ACM Trans. Graph.* 26, 3 (2007), 99.

Received January 2017; final version April 2017; accepted April 2017